Milestone 3 report

Name	ID
Adel Mahmoud Mohamed Abdelrahman	20010769
Mohamed Hassan Sadek Abdelhamid	20011539
Mohamed Hassan Yousef Hassan Eltobgy	20011544
Mohamed Raffeek Mohamed El-Sayed	20011574
Mahmoud Attia Mohamed Abdelaziz Zian	20011810
Mahmoud Ali Ahmed Ali Ghallab	20011811

Requirements:

Root admin adds new admins
Function: The system should allow the root admin to add new administrators.
Description: • As a root admin, I expect to be able to add new admins. I expect to find a button to add a new admin in the user drop down list, when clicking on the button a popup window appears to enter his email and the response then is viewed to me (success/fail). • An invitation mail is sent to him/her, through it, he/she can verify email and complete sign up as an admin entering the password, first name, and last name and submit this by clicking on the form signup button. the response is viewed to the new admin and in case of success sign up he/she will be navigated to the home page.
Inputs: • Email of the new admin.
Source: • Root admin initiates the process through a button in the user drop-down list.
Outputs: Response message (success (green colored message)/fail (red colored message)) after attempting to add a new admin is viewed in the pop window.
Destination: Displayed to the root admin.
Action: When the root admin clicks the "Add Admin" button, a popup window appears to input the new admin's email. After entering the email, the system processes the request.
Requirements: root admin is saved in database.
Pre-condition: The root admin is logged in.

Post-Condition: The system displays a success/fail response to the root admin.

Side effects: the unverified admin is stored in the database waiting to complete register and verify.

Additional Actions:

- Upon success, an invitation email is sent to the new admin.
- The new admin verifies their email and completes the signup process.
- Signup form inputs: Password, first name, last name (and email is viewed but with disabled edit).
- Response: Success or failure of the signup process.
- If successful, the new admin is navigated to the home page.

Customer reviewing a product
Function: review a product
Description: each user can express his opinion (review) about a product and leave a star rating (1-5)
Inputs: • User actions: Writing a review, submitting a rating (1-5 stars)
Source: • Add Review button: when clicked, a text box and initially empty stars appears where the user can write his review and choose the rating.
Outputs: 1. The review and rating are displayed on the product details page for other customers to see. 2. The aggregate rating of the product may be updated based on the new review. 3. User feedback is stored in the database for future reference and analysis.
Destination: Upon submitting a review, the user's input is processed by the system and stored in the database. The review and updated rating are then displayed on the product details page.
Action: • Customer clicks on “Add Review” generating a text box and empty stars along with 2 buttons (cancel, Post Review) • if customer presses cancel the generated form will be hidden. • if customer presses Post Review, the review will be stored in database if and only if the star rating and the review text both are not empty.
Requirements: product must exist and is saved in database
Pre-condition: User must be a registered customer (admins can only view reviews, they can't add or edit it)
Post-Condition: none
Side effects: UI is updated

Search Products
Function: The user can use the search bar in the top navbar to search on products
Description: As a user (customer/admin), I expect to find a search bar in the top navbar, that is accessible in all pages. I can use this search bar to enter some keyword to search on the products that match this keyword.

Inputs: Keyword to search on. The keyword can be at maximum of 200 characters. It can contain numbers, letters, and icons.
Source: User types a keyword in the search bar.
Outputs: List of products that matches this keyword. The list is displayed as products are displayed in the products catalog. The matching can occur in the product name, category, brand, or description. The matching in the details of the product doesn't have to be exact. If the keyword is present in any part in any pattern of the 4 product details mentioned, this product will be included in the response list.
Destination: The system to process the keywords and finds the matching products.
Action: When the user types a keyword in the search bar, and then types on the search icon, the user is directed to the product catalog page with only the matching products displayed on the page.
Requirements: product must exist and is saved in database
Pre-condition: The user is logged in.
Post-Condition: The system displays a success/fail response to the root admin.
Side effects: The user is redirected to the product catalog page

Filter Products
Function: Filter the list of products
Description: products are filtered based on specific keys (range of prices, names, description, brand, category, …)
Inputs: range of prices, product name, description, brand, name and category
Source: text boxes, sliders
Outputs: List of products that matches the specifications chosen will be displayed.
Destination: The resulted list of products is displayed in the same page
Action: When the user chooses arbitrary specifications all products will disappear leaving only products of satisfying the specifications chosen.
Requirements: input to filter by must exist
Pre-condition: there must be products
Post-Condition: only products satisfying chosen specifications will appear
Side effects: NA

Filter Orders
Function: Order Filtering in Vendor Dashboard.
Description: Vendors should have the ability to filter orders based on order ID, customer ID, and the overall price range.
Inputs: 1. Vendor-selected filter criteria: a. order ID

b. customer ID

c. price range

d. Minimum and maximum price values for the price range.

Source: Vendor input through the dashboard.

Outputs: Filtered order data for display in the dashboard.

Destination: Vendor dashboard.

Action:

1. The vendor interacts with filter options on the dashboard.
2. The backend processes the filtering criteria and retrieves relevant data from the database.

Requirements:

1. The front-end shall provide filter options (checkboxes, dropdowns) for order ID, customer ID, and price range.
2. The front-end shall dynamically update the order listings based on the selected filter criteria without requiring a full page reload.
3. The backend shall have a controller in the controllers directory to handle search and filter requests.
4. The controller shall parse incoming requests to extract search keywords, filter criteria, and the entity to filter (orders).
5. The controller shall call the relevant service to retrieve filtered order data from the database.
6. In the services directory, a service shall handle the search and filter logic for orders.
7. The service shall return the filtered order data to the controller.
8. The controller shall send the filtered order data back to the front-end for display.
9. The backend shall support filtering based on order ID, customer ID, and an overall price range.
10. The backend shall handle the inclusion of minimum and maximum price values for the price range.
11. The frontend shall provide a user-friendly interface for vendors to interact with filters.

Pre-condition:

1. Vendors must be authorized / logged in.
2. Vendors are in the dashboard.
3. Vendors must have at least one product.

Post-condition:

The vendor sees a dynamically updated list of orders based on the selected filter criteria.

Side effects:

the orders display by UI and retrieval from database according to the specific option.

Sort Products
Function: Sort the list of products
Description: The products are sorted given some sorting criteria and sorting order.
Inputs: The sorting criteria from dropdown list and the sorting order which is whether ascending or descending order chosen from another dropdown list. The sorting criteria can be one of these: product name, price, average users rating, number of reviews, publish date, remaining count in stock, number of sold items,
Source: The user choses the sorting criteria and sorting order from the dropdown list.

Outputs: List of sorted products viewed on the product catalog page.

Destination: Product catalog page.

Action: The user first goes to the product catalog page. Then he clicks on the sort button. A pop-up window opens to choose the sorting criteria and sorting type. Then he clicks on the apply button to confirm his input.

Requirements: The user must be logged in first.

Pre-condition: NA

Post-condition: NA

Side effects: The page reloads after sorting to view the new products listing.

Sort Orders
Function: Order Sorting in Dashboard
Description: Vendors should have the ability to sort orders in ascending or descending order based on various attributes. Vendor-selected sorting criteria and order (ascending/descending). 1. Sorting options for 2. order ID 3. order Name 4. Price 5. Rating 6. Reviews 7. Date Added 8. Remaining in Stock 9. Sold Count 10. Brand 11. delivery date
Inputs: Vendor-selected sorting criteria and order (ascending/descending). Sorting options for Sorting options for order ID, order Name, Price, Rating, Reviews, Date Added, Remaining in Stock, Sold Count, Brand and delivery date
Source: Vendor input through the dashboard.
Outputs: Sorted order data for display in the dashboard.
Destination: Vendor dashboard page.
Action: 1. The vendor interacts with sorting options on the dashboard. 2. The backend processes the sorting criteria and retrieves relevant data from the database.
Requirements: 1. The front-end shall provide sorting options (checkboxes, dropdowns) for order ID, order Name, Price, Rating, Reviews, Date Added, Remaining in Stock, Sold Count, Brand and delivery date.

2. The front-end shall include a sorting functionality to enable vendors to choose between ascending and descending order.
3. The backend shall have a controller in the controllers directory to handle incoming sort requests.
4. The controller shall parse incoming requests to extract sort type (ascending/descending) and criteria.
5. The controller shall call the relevant service to retrieve sorted order data from the database.
6. In the services directory, a service shall handle the sort logic for orders.
7. The service shall return the sorted order data to the controller.
8. The controller shall send the sorted order data back to the front-end for display.
9. The backend shall support sorting based on order ID, customer ID, overall cost, order date, delivery date, address, and total items.
10. The frontend shall provide a user-friendly interface for vendors to interact with sorting options.

Pre-condition: Vendors must be logged in and authenticated.

Post-condition: The vendor sees a dynamically sorted list of orders based on the selected criteria.

Side effects: The orders sorted based on the specific category.

Category Page

Function: View, Add, and Edit Categories

Description: Users, depending on their roles, should be able to view a list of categories. Admins have the additional capability to add and edit categories.

Inputs:

- For Admin:
  - Category information (name, imageUrl) for adding/editing.
- For Customer:
  - No direct inputs for adding/editing.

Source:

- Admin: Admin input and system data.
- Customer: System data.

Outputs:

- Displayed list of categories.
- Success/failure messages for add/edit operations for admin.

Destination: Category page for both Admin and Customer.

Action:

- Admin can view, add or edit categories.
- Customer can only view categories.

Requirements:

1. The system shall provide a Category Page including its products accessible to both Admin and Customer.
2. The Category Page for Admin shall include options to add and edit categories.
3. The Category Page for Customer shall display a list of all available categories without options to add/edit.
4. Admin input for adding/editing categories shall include category name and details.
5. The system shall validate and handle errors for category add/edit operations.
6. Success or failure messages shall be displayed after add/edit operations.

7. The Category Page shall be responsive and user-friendly for easy navigation.

Pre-condition:

- Users (Customers, Admins) must be logged in/signed up.
- Admin must have appropriate permissions to add/edit categories.

Post-condition:

- The list of categories is updated based on successful add/edit operations.
- Messages are displayed to indicate the success or failure of category add/edit.

Side effects: products of chosen category will appear.

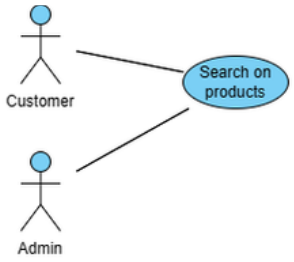
Home Page
Function: Display Personalized Product Recommendations
Description: Creating a personalized product recommendations based on the latest, most sold, and most popular products, and the user can easily add any product in the recommended products sections to his cart or wishlist, and the admin can edit both the displayed catergories and products.
Inputs: User visits the main page or a product-specific page.
Source: User action triggers the recommendation system.
Outputs: A visually appealing section displays products that are on offer or are bestsellers. Each product includes brief and informative descriptions such as its name, price, image, and rating. And the categories are displayed as well
Destination: The personalized product recommendations are displayed in a dedicated section on the main page or product-specific page.
Action: The system analyzes user behavior and preferences to dynamically generate personalized product recommendations.
Requirements: The recommendation engine must be implemented, and product as well as the categories information must be available in the system.
Pre-condition: The user visits the main page or a product-specific page.
Post-Condition: The personalized product recommendations are displayed based on user behavior.
Side effects: Enhanced user engagement and increased likelihood of product discovery.

Customer Order History
Function: save history of orders made by a customer
Description: The menu should contain all orders the user made. Each order is displayed in the list showing the date the order was created, the total cost of the order, and the status of the order which should be one of these: pending, shipped, delivered, or canceled.
Inputs: previously made orders
Source: orders loaded from database
Outputs: • list of orders

• list of products withing orders when an order is chosen
Destination: orders web page
Action: • Each order is displayed in the list showing the date the order was created, the total cost of the order, and the status of the order which should be one of these: pending, shipped, delivered, or canceled. • When clicking on an order a list of products is shown, each product item has a name, image, number of items purchased, and the price of one item of this product. At the end of this page, the total cost is displayed.
Requirements: User must be registered
Pre-condition: Customer made orders previously
Post-Condition: orders are shown or nothing is shown if there are no previous orders
Side effects: none

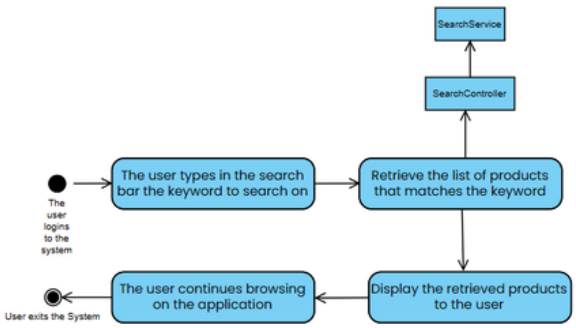
Diagrams:

- Search use case diagram (use case)

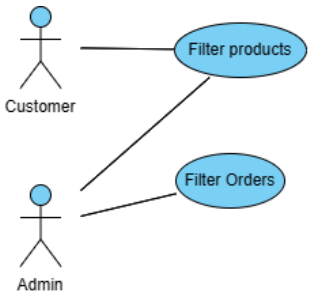


Our system users are given the opportunity to search using a keyword and then find any products that match this keyword.
The keyword may be part of the product name, brand, category, or description

- Search activity diagram (activity diagram)



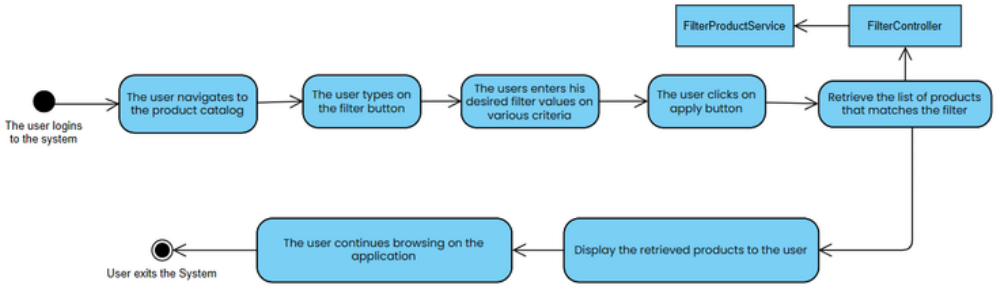
- Filter Products/Orders use case diagram(use case)



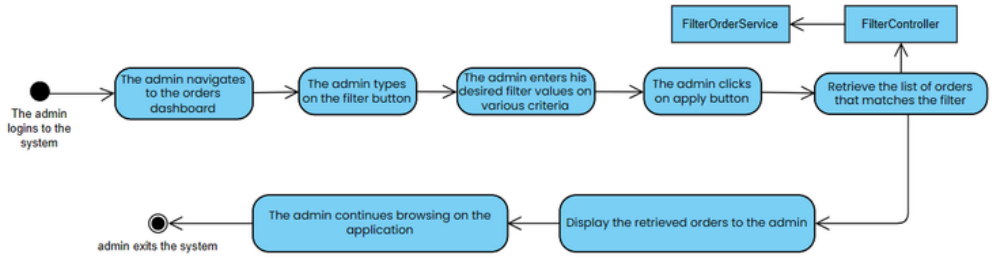
Our users are able to filter the products according to many criteria at the same time to help them find what they want faster. They can filter based on: product name, brand, category, description, price range, discount range, or include out of stock or not.

The admin in our system is able to filter the orders in his dashboard according to order id, customer id, total cost range, or status, so that he can find the desirable orders faster.

- Filter Products activity diagram (activity diagram)

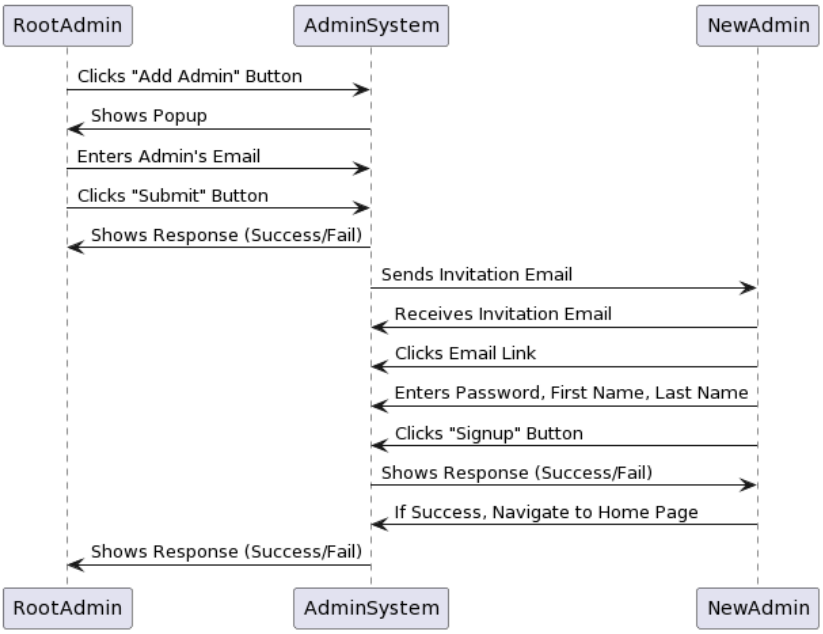


- Filter Orders activity diagram (activity diagram)



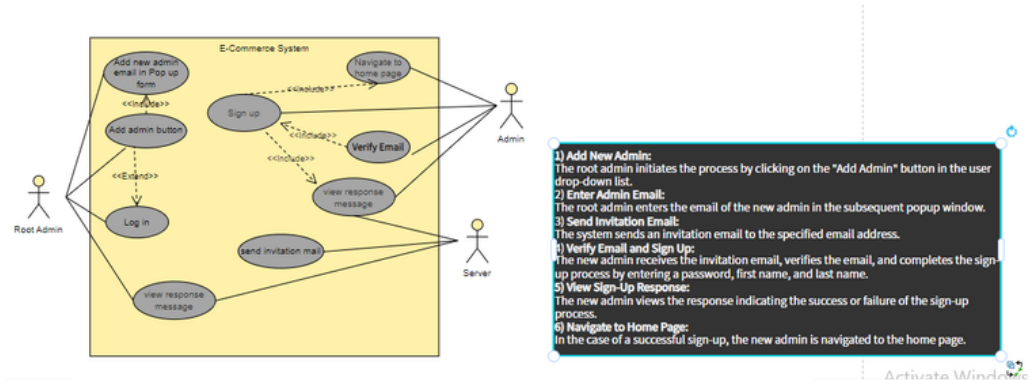
- Root Admin adding new admin sequence diagram:

RootAdmin adds New Admins sequence diagram.png



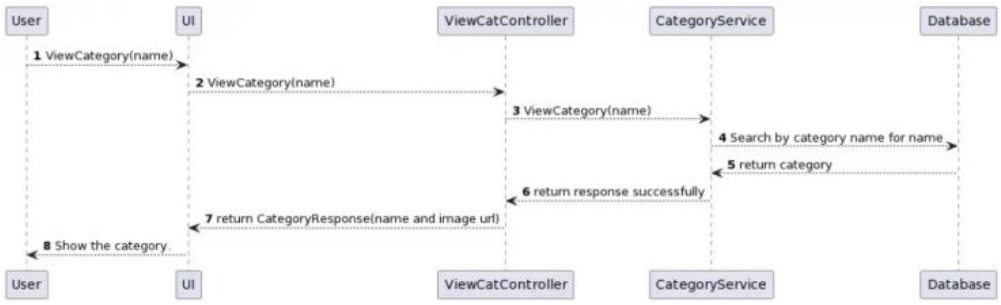
- Root Admin adding new admin use case diagram:

Root Admin adds admin use case.png

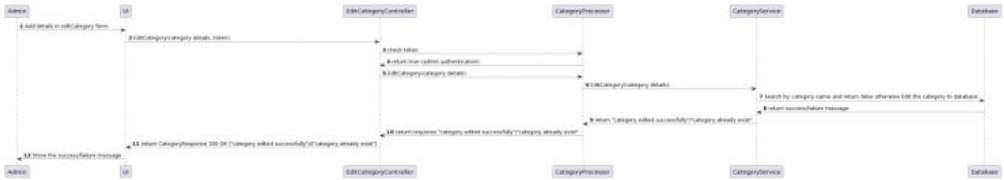


- **Category list Sequence diagrams:**

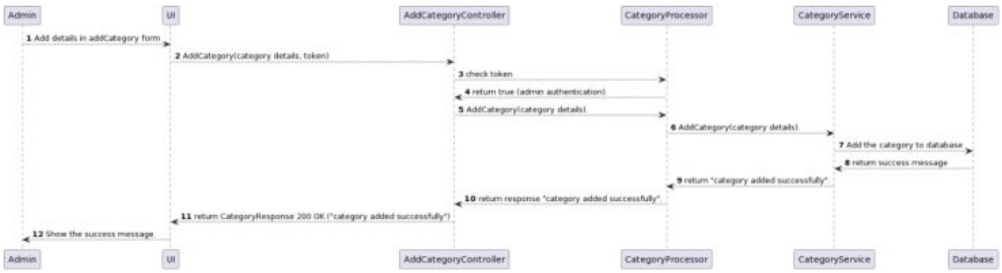
View Category SeqDiagram.webp



Edit Category Seq diagram.webp

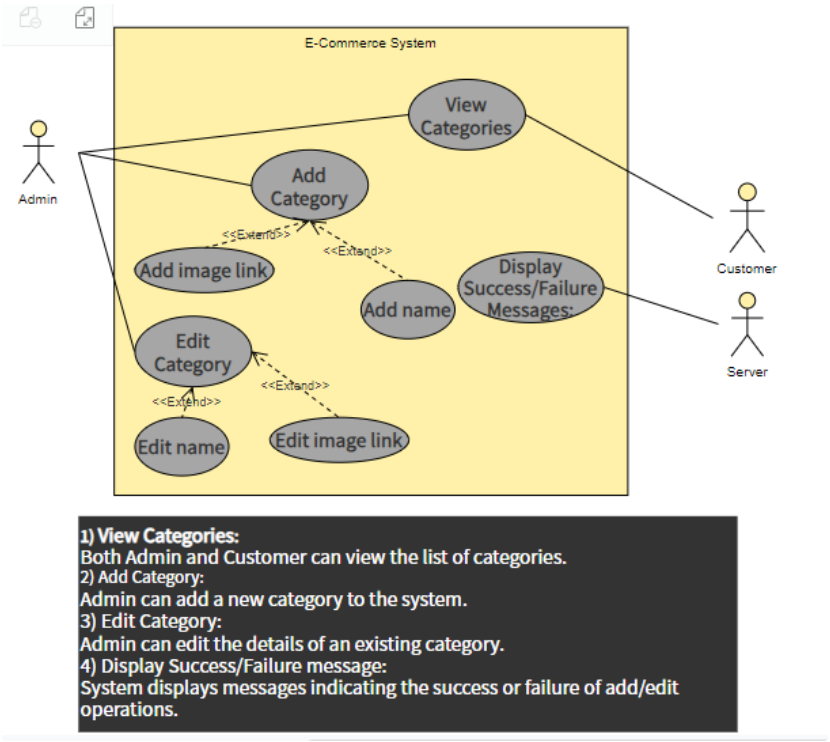


Add Category sequence diagram.webp



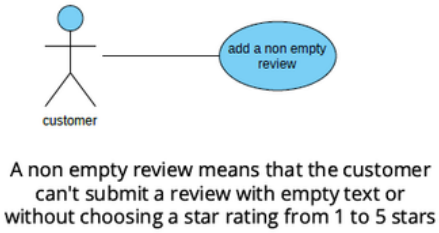
- **Category list use case diagrams:**

Category List.png



Customer reviews Use case diagram:

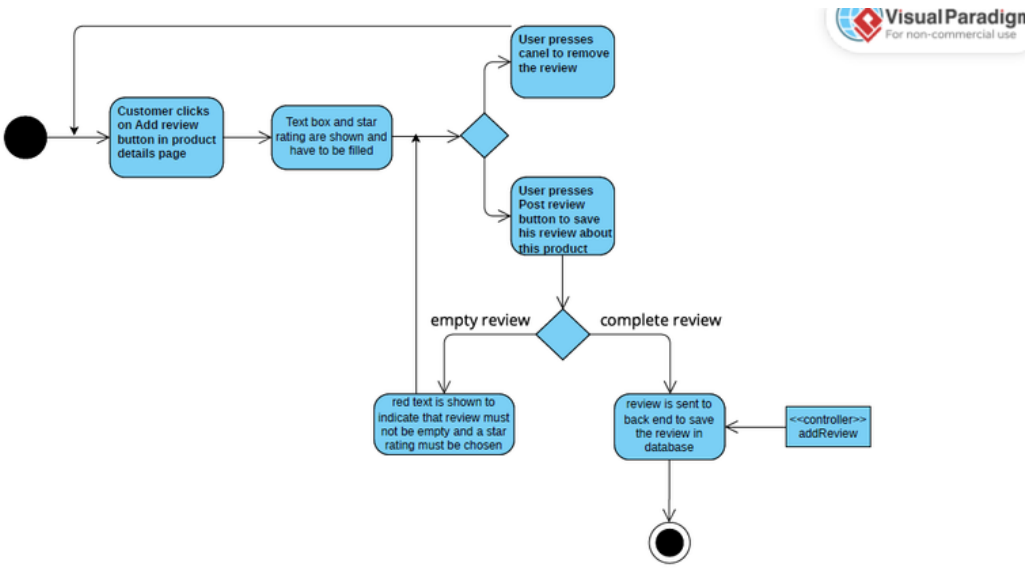
Customer Reviews Use case diagram.pdf



A non empty review means that the customer can't submit a review with empty text or without choosing a star rating from 1 to 5 stars

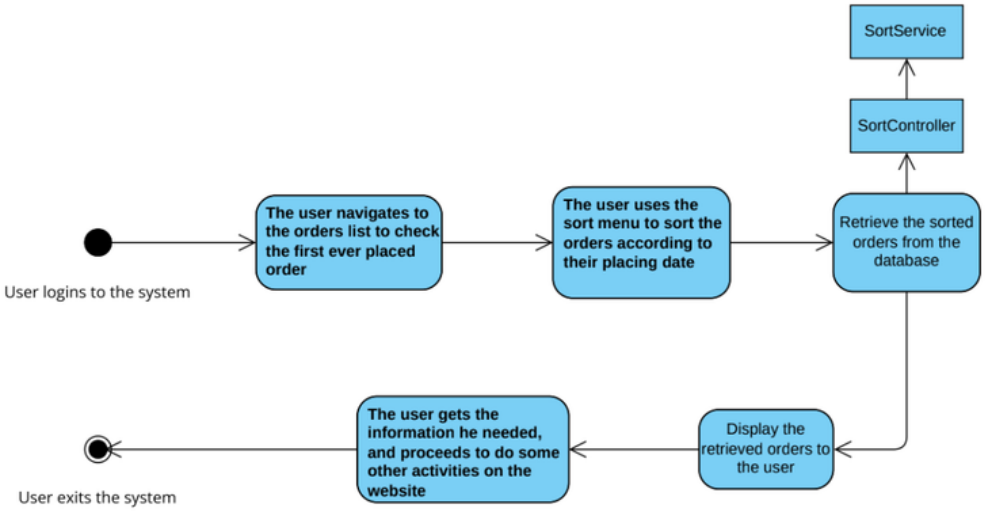
Customer reviews Activity diagram:

Customer Review Activity Diagram.pdf



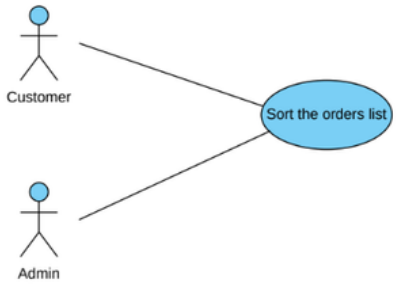
Order sort Activity Diagram:

Orders sorting activity diagram.pdf



Order sort Use case Diagram:

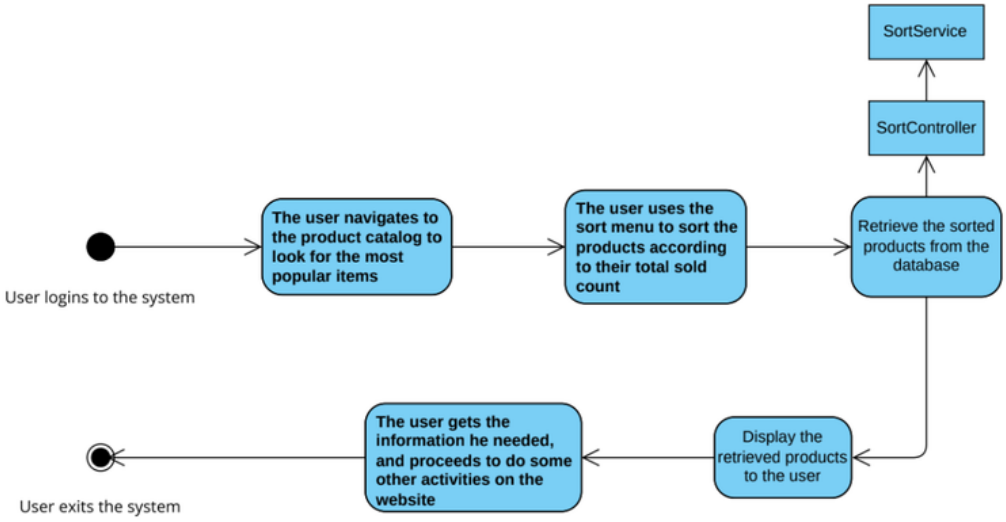
Orders sort use case.pdf



Our system users are given the option to seaminglessly sort the orders list to make it easier for them to reach a specific set of orders. They can sort the orders according to some of the attributes, such as order id, order placing date, total cost, and much more through a user friendly UI.

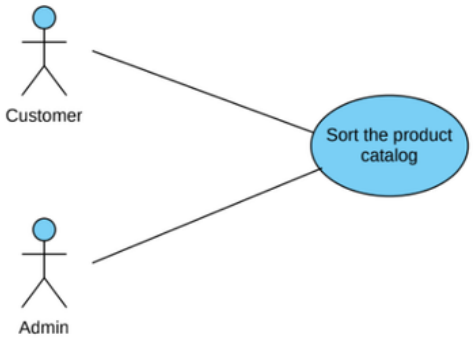
Product sorting Activity Diagram:

Products sorting activity diagram.pdf



Product sorting Use case Diagram:

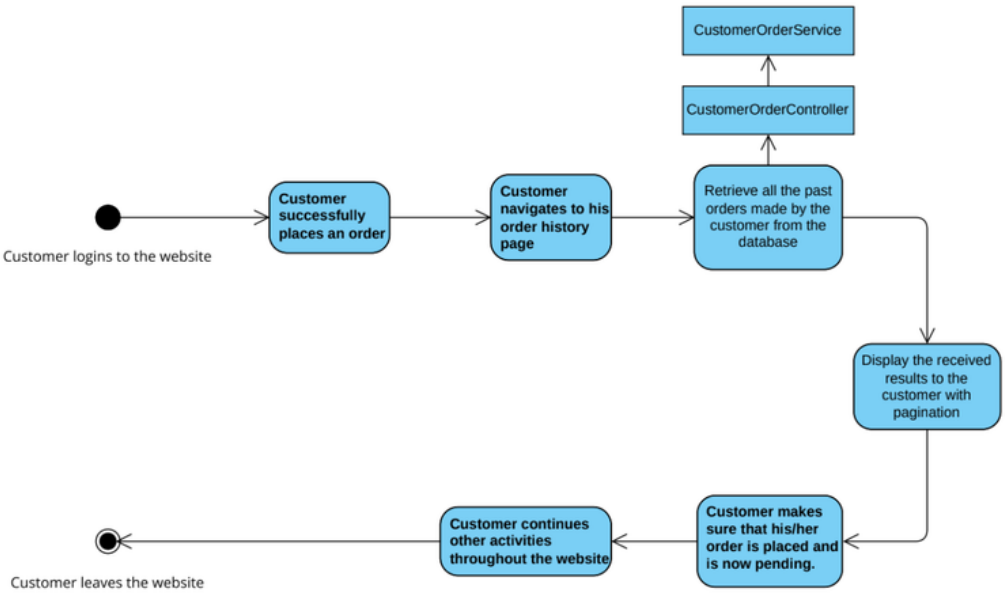
Product sorting use case.pdf



Our system users are given the option to seaminglessly sort the products' catalog to make it easier for them to reach a specific set of products. They can sort the products according to some of the attributes, such as product name, price, popularity (through sold count), ratings, and much more through a user friendly UI.

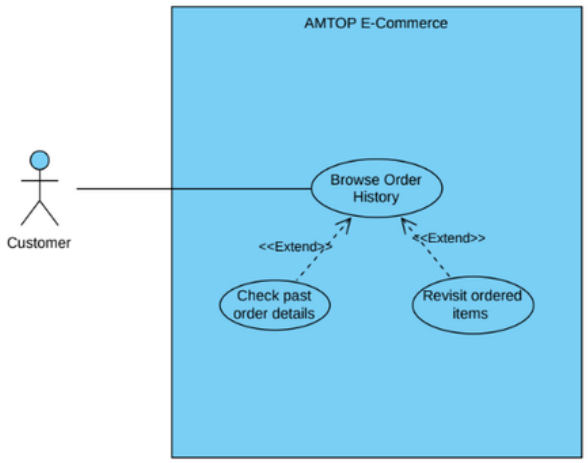
Customer orders Activity Diagram:

Customer orders activity diagram.pdf



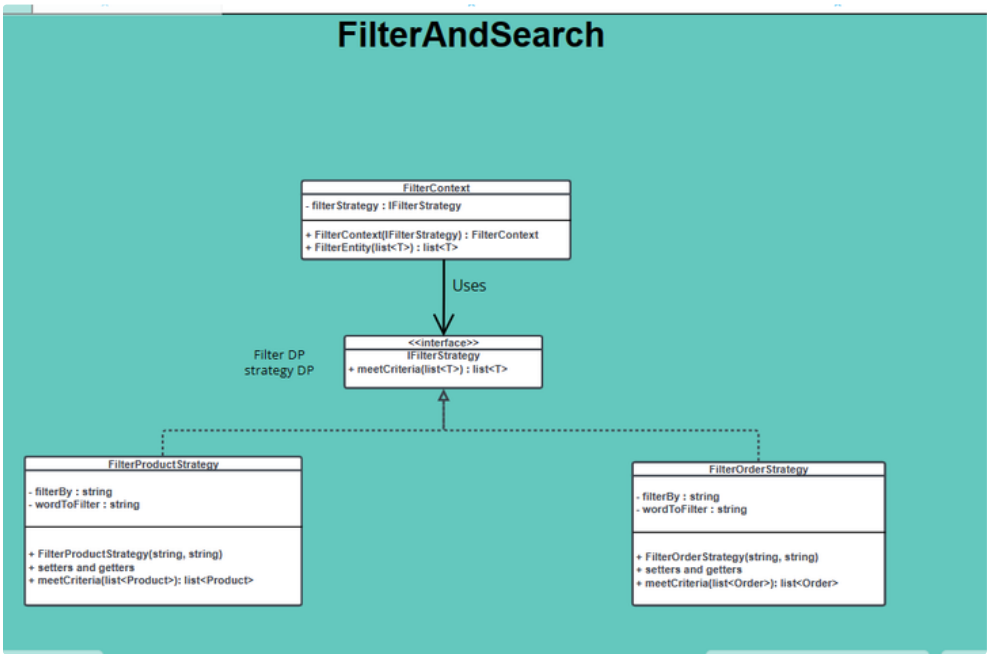
Customer orders Use case Diagram:

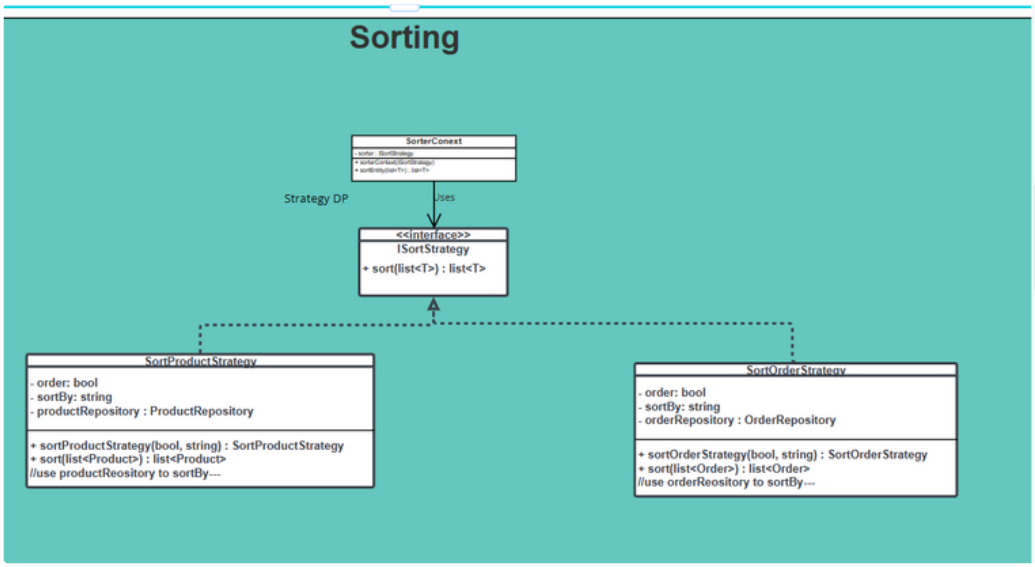
Customer orders use case.pdf



The customer is given the ability to browse his past orders seeminglessly. He/She can check past order details such as total price, order items, order status, and much more through a user friendly interface.

Design changes:





Git Branching Model: